Vysoká škola ekonomická v Praze

Fakulta informatiky a statistiky

****

Vliv renderingu webových aplikací na web vitals metrics a SEO

diplomová PRÁCE

Studijní program: Aplikovaná informatika

Studijní obor: Znalostní a webové technologie

Autor: Bc. Lukáš Březina

Vedoucí diplomové práce: Ing. et Ing. Stanislav Vojíř, Ph.D.

Konzultant diplomové práce: [jméno a příjmení vedoucího vč. titulů]

Praha, květen 2021

Prohlášení

Prohlašuji, že jsem bakalářskou práci [Název práce] [vypracoval/vypracovala] samostatně za použití v práci uvedených pramenů a literatury.

V Praze dne [datum vč. roku (měsíc napsat slovy)]

[jméno a příjmení autora]

Poděkování

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean id metus id velit ullamcorper pulvinar. In convallis. Sed convallis magna eu sem. Maecenas lorem. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Ut tempus purus at lorem. Phasellus et lorem id felis nonummy placerat. Mauris elementum mauris vitae tortor. Aliquam id dolor. Vivamus porttitor turpis ac leo.

Abstrakt

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean id metus id velit ullamcorper pulvinar. In convallis. Sed convallis magna eu sem. Maecenas lorem. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Ut tempus purus at lorem. Phasellus et lorem id felis nonummy placerat. Mauris elementum mauris vitae tortor. Aliquam id dolor. Vivamus porttitor turpis ac leo.

Nullam justo enim, consectetuer nec, ullamcorper ac, vestibulum in, elit. Integer vulputate sem a nibh rutrum consequat. Temporibus autem quibusdam et aut officiis debitis aut rerum necessitatibus saepe eveniet ut et voluptates repudiandae sint et molestiae non recusandae. Ut tempus purus at lorem. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam.

Maecenas fermentum, sem in pharetra pellentesque, velit turpis volutpat ante, in pharetra metus odio a lectus. Vivamus ac leo pretium faucibus. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nullam dapibus fermentum ipsum. In sem justo, commodo ut, suscipit at, pharetra vitae, orci. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit.

Klíčová slova

amet, consectetuer, dolor, Lorem ipsum, sit.

JEL klasifikace

amet, consectetuer, dolor, Lorem ipsum, sit.

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean id metus id velit ullamcorper pulvinar. In convallis. Sed convallis magna eu sem. Maecenas lorem. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Ut tempus purus at lorem. Phasellus et lorem id felis nonummy placerat. Mauris elementum mauris vitae tortor. Aliquam id dolor. Vivamus porttitor turpis ac leo.

Nullam justo enim, consectetuer nec, ullamcorper ac, vestibulum in, elit. Integer vulputate sem a nibh rutrum consequat. Temporibus autem quibusdam et aut officiis debitis aut rerum necessitatibus saepe eveniet ut et voluptates repudiandae sint et molestiae non recusandae. Ut tempus purus at lorem. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam.

Maecenas fermentum, sem in pharetra pellentesque, velit turpis volutpat ante, in pharetra metus odio a lectus. Vivamus ac leo pretium faucibus. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nullam dapibus fermentum ipsum. In sem justo, commodo ut, suscipit at, pharetra vitae, orci. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit.

Keywords

amet, consectetuer, dolor, Lorem ipsum, sit.

JEL Classification

amet, consectetuer, dolor, Lorem ipsum, sit.

Obsah

[Úvod 11](#_Toc530041136)

[1 Obrázky, tabulky a programy 12](#_Toc530041137)

[1.1 Obrázky 12](#_Toc530041138)

[1.1.1 Tabulky 13](#_Toc530041139)

[1.1.2 Programy 14](#_Toc530041140)

[1.1.3 Sazba matematiky 15](#_Toc530041141)

[2 Práce s literaturou 16](#_Toc530041142)

[Závěr 17](#_Toc530041143)

[Použitá literatura 18](#_Toc530041144)

[Přílohy I](#_Toc530041145)

[Příloha A: Název první přílohy I](#_Toc530041146)

[Příloha B: Název druhé přílohy I](#_Toc530041147)

Seznam obrázků

[Obr. 1.1 Četnosti různých velikostí bot v populaci žen, mužů a dětí (vlastní zpracování) 13](#_Toc529974559)

Poznámka: Seznam obrázků je vhodný použít, pokud počet obrázků v textu práce je větší než 20. Seznam grafů je vhodný použít pouze v případě, že autor práce rozlišuje mezi obrázkem a grafem. Seznam grafů je utvářen, pokud je počet grafů větší než 20. V této šabloně závěrečné práce se grafy a obrázky nerozlišují.

Seznam tabulek

[Tab. 1.1 Záznam výsledků závodu robotů (vlastní zpracování) 14](#_Toc529974579)

Poznámka: Seznam tabulek je vhodný použít, pokud počet tabulek v textu práce je větší než 20.

Seznam výpisů programového kódu

[Výpis 1.1 Ukázka formátování krátkého výpisu programového kódu (vlastní zpracování) 14](#_Toc529974597)

Poznámka: Seznam výpisů programového kódu je vhodný použít, pokud počet vložených objektů tohoto typu je větší než 20.

Seznam zkratek

SSR server-side rendering

SPA single page application

API

DOM

HTML

CSS

JS

DP diplomová práce

FIS Fakulta informatiky a statistiky

UNESCO Organizace OSN pro vzdělání, vědu a kulturu

NATO Severoatlantická aliance

Poznámka: Seznam zkratek je vhodný použít, pokud počet zkratek v textu práce je větší než 20 a nejedná se o zkratky běžné.

Úvod

V diplomové je rozebrán vliv různých druhů renderingu webových aplikací na jejich výkonnost. Kromě toho je snaha popsat, jak si s jednotlivými druhy renderingu umí poradit vyhledávací a jiní roboti a jaký vliv mají tyto druhy renderingu na SEO.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean id metus id velit ullamcorper pulvinar. In convallis. Sed convallis magna eu sem. Maecenas lorem. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Ut tempus purus at lorem. Phasellus et lorem id felis nonummy placerat. Mauris elementum mauris vitae tortor. Aliquam id dolor. Vivamus porttitor turpis ac leo. Maecenas fermentum, sem in pharetra pellentesque, velit turpis volutpat ante, in pharetra metus odio a lectus. Phasellus et lorem id felis nonummy placerat.

# Teoretická analýza

V následující kapitole je popsán teoretický základ, na kterých je tato práce postavena. V první části se rozebírá, jaké různé druhy renderingu se v dnešních webových aplikacích používají. Druhá část se zaměřuje na měřitelnost výkonu webových aplikací. Ve třetí části je popsán aktuální stav vybraných vyhledávacích enginů a jak tyto enginy s různými druhy renderingu zacházejí.

## Druhy renderingu

Ze začátku je potřeba vyjasnit, o jakém renderingu webových aplikací tato práce pojednává. Renderingem webové aplikace v kontexu této práce je myšleno především převedení surových dat do HTML a CSS. Explicitně se tedy nejedná o rendering, který řeší vykreslovací jádra jednotlivých prohlížečů – tj. převod HTML a CSS na jednotlivé pixely. V práci je tedy řešeno, v jaký čas a čími výpočetními prostředky převedeme surová data do formy, kterou již umí vykreslovací jádra vykreslit (HTML, CSS).

V následující sekci jsou popsány jednotlivé přístupy k renderingu. Je vycházeno především ze článku (Miller a Osmani, 2019), kde jsou jednotlivé druhy velice dobře popsány a shrnuty. U každého druhu je rozebrána typická architektura aplikace rendering využívající a princip fungování. Je naznačeno, jak se u jednotlivých druhů renderingu pracuje s daty a jak vypadá odpověď na HTTP požadavek pro jednooduchou aplikaci úkolníčku. Také je u každého druhu dán za příklad framework, knihovna, či aplikace, tento druh renderingu používající.

### Server rendering

Jedná se o nejstarší, nejklasičtější a také nejrozšířenější podobu renderingu webových aplikací. Prohlížeč na HTTP požadavek obdrží v odpovědi HTML dokument již obsahující data, která chceme vykreslit, také ve formátu HTML (viz Výpis 1.1 Nástin HTTP odpovědi pro server-rendered stránku úkolníčku). V rámci tohoto dokumentu jsou také v hlavičkách obvykle připojeny CSS upravující vzhled stránky. Javascript se při tomto druhu renderingu obvykle používá hlavně pro „zinteraktivnění“ aplikace – např. validování uživatelských vstupů ve formuláři, pokročilé animace, drag&drop operace atd. Serverově renderované aplikace jsou také specifické tím, že jejich stav (např. položky v košíku) se obvykle drží kompletně na serveru. Změna tohoto stavu probíhá obvykle posláním příslušného HTTP požadavku, který opět vrátí kompletní HTML dokument zobrazující již aktualizovaný stav.

Z hlediska softwarové architektury jsou serverově renderované aplikace obvykle monolitického typu[[1]](#footnote-2). Jedná se tedy o jedinou aplikaci řešící vše od zpracování HTTP požadavku, přístupu k databázi, až po vyrenderování samotného HTML.

Na serverově renderované aplikace existuje nespočet frameworků a jazyků, ve kterých je možno je psát. Nejčastěji používaným je jazyk PHP. Zástupcem serverově renderované aplikace může být například náš školní informační systém[[2]](#footnote-3).

Výpis 1.1 Nástin HTTP odpovědi pro server-rendered stránku úkolníčku

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="stylesheet" href="style.css">

<title>Úkolníček</title>

</head>

<body>

<h1>Výpis úkolů</h1>

<ul>

<li><a href="/ukol/1">Vyvenčit psa</a></li>

<li><a href="/ukol/2">Vynést koše</a></li>

</ul>

<form method="POST" action="/pridat\_ukol">

<label for="ukol">Úkol</label>

<input id="ukol" type="text">

<button>Přidat</button>

</form>

<script src="script.js" type="text/javascript"></script>

</body>

</html>

### Client-side rendering

Na rozdíl od server-rendered aplikací obvykle veškeré HTTP požadavky na danou aplikaci vrací stejný obsah. Jedná se o HTML dokument obsahující odkazy na CSS a JS, který obvykle v těle obsahuje pouze text, popř. animaci, vizualizující načítání aplikace (viz Výpis 1.2 Nástin HTTP odpovědi pro client-rendered stránku úkolníčku). Jak a odkud se vezmou potřebná data pro daný požadavek, či jak se tyto data vykreslí, řeší již javascriptová aplikace odkazovaná v HTML dokumentu, u klienta v prohlížeči. Obvykle tedy javascript po načtení odešle několik HTTP požadavků na API, ze kterých získá potřebná data. Ty poté vykreslí do DOM (viz Výpis 1.3 client-side rendered stránka úkolníčku). Dost často tyto aplikace také drží velkou část stavu aplikace. Změna stavu obvykle probíhá tak, že se na pozadí pošle HTTP požadavek o změnu stavu a v případě úspěchu se překreslí pouze části aplikace ovlivněné změnou tohoto stavu (např. změna čísla s počtem položek v košíku v hlavičce po odebrání z košíku). Hlavní výhodou aplikací renderovaných na klientské straně je tedy absence kompletního přenačítávání celé stránky s každou změnou stavu. Zásadní nevýhodou je pomalé první načtení stránky.

Z architektonického hlediska se obvykle jedná o client-server architekturu[[3]](#footnote-4). Webová aplikace má obvykle dvě oddělené části – tzv. backend (server) a frontend (client), které mezi sebou komunikují pomocí API. Backendová část obvykle řeší ukládání a zpřístupnění dat, náročnější business logiku, komunikaci s jinými třetími stranami, či výpočetně náročné operace. Frontendová část tvoří uživatelské rozhraní aplikace a má na starosti rendering. Uživatel celou dobu interaguje pouze s touto částí, která jeho akce převádí na jednotlivá API volání. Velkou výhodou této architektury webových aplikací je to, že můžeme v tomto případě podle potřeby používat a libovolně kombinovat více backendových částí (např. aplikace sdružující více informačních systémů), více frontendových částí (web, mobilní aplikace, televizní aplikace) či různé technologie/architektury pro API (REST[[4]](#footnote-5), GraphQL[[5]](#footnote-6), Websocket[[6]](#footnote-7)). V případě frontend části se stala standartem component-oriented architektura[[7]](#footnote-8), která se také velmi liší od architektury používané v monolitických serverově renderovaných aplikací. Základem jsou znovu použitelné malé celky skládající se z šablony, stylů a logiky – komponenty.

Nejznámější frameworky/knihovny využívající client-side rendering jsou React[[8]](#footnote-9), Angular [[9]](#footnote-10)a Vue[[10]](#footnote-11). Typickou client-side rendered aplikací je např. Gmail[[11]](#footnote-12).

Výpis 1.2 Nástin HTTP odpovědi pro client-rendered stránku úkolníčku

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="stylesheet" href="style.css">

<script src="app.js" type="text/javascript"></script>

<title>Úkolníček</title>

</head>

<body>

<app-root>

<img src=“nacitani.gif“ alt=“Načítám“>

</body>

</html>

Výpis 1.3 client-side rendered stránka úkolníčku

…

<body>

<app-root>

<app-vypis-ukolu>

<h1>Výpis úkolů</h1>

<ul>

<!-- data loaded from API, redirect on link click

prevented and handled by JS -->

<li><a href="/ukol/1" routerLink="ukol/1">Vyvenčit psa</a></li>

<li><a href="/ukol/2" routerLink="ukol/1">Vynést koše</a></li>

</ul>

</app-vypis-ukolu>

<app-novy-ukol>

<!-- submit and sending data handled by JS -->

<form>

<label for="ukol">Úkol</label>

<input id="ukol" type="text">

<button>Přidat</button>

</form>

</app-novy-ukol>

</app-root>

</body>

…

### Client-side rendering with prerendering

Jedná se o jeden z přístupů, jak zlepšit rychlost prvního načtení stránky. Aplikace je psána stejným způsobem, jako client-side rendered aplikaci. Building proces je ale upraven tak, že negeneruje prázdný HTML dokument, ale již předgenerovanou HTML strukturu (viz Výpis 1.4 Nástin HTTP odpovědi pro CSR with prerenering stránku úkolníčku), do které se po načtení javascriptu pouze načtou dynamická data a stránka vypadá stejně, jako by byla vyrenderovaná na klientské straně (viz Výpis 1.3 client-side rendered stránka úkolníčku.

Výpis 1.4 Nástin HTTP odpovědi pro CSR with prerenering stránku úkolníčku

…

<body>

<app-root>

<app-vypis-ukolu>

<h1>Výpis úkolů</h1>

<ul>

</ul>

</app-vypis-ukolu>

<app-novy-ukol>

<!— submit nic neudělá dokud nenastartuje js aplikace -->

<form>

<label for="ukol">Úkol</label>

<input id="ukol" type="text">

<button>Přidat</button>

</form>

</app-novy-ukol>

</app-root>

</body>

…

### Server-side rendering with (re)hydration

Javascript může být kromě prohlížeče spouštěn také na serveru díky NodeJS[[12]](#footnote-13). Tohoto faktu využívá právě server-side rendering. Server-side renderingem rozumíme spuštění kódu client-side rendered aplikace pro daný požadavek na serveru. Výstup tohoto spuštění je tedy HTML dokument včetně stylů, který odpovídá tomu, co vyrenderuje klientská aplikace po načtení. Problém nastává, když výstup tohoto HTML dokumentu zobrazí prohlížeč. Stránka sice vypadá až na výjimky identicky jako client-side vyrenderovaná, ale není interaktivní – javascriptový kód registrující event listeners byl již spuštěn na serveru. „Zinteraktivnění“ takto vygenerovaného dokumentu se nazývá hydration. Několik způsobů, jak tuto hydration provést popíšeme v následujících odstavcích.

#### Rehydration

První možností je tzv. rehydration. Součástí dokumentu vráceným serverem je kromě onoho statického HTML také skript klientské aplikace a serializovaná data, která byla použita pro její rendering. Prohlížeč tedy HTML dokument vykreslí a přibalený skript se spustí jako při klasicky klientsky renderované aplikaci. Protože už má dostupná všechna data, která by obvykle získával voláním např. XmlHttpRequestu v serializované podobě, může klient renderovat aplikaci velmi rychle. Po dokončení se buď serverem renderovaný DOM kompletně zahodí a nahradí DOMem vyrenderovaný klientem. Druhá možnost je, že se DOM nezahazuje, ale na již existující DOM se napojí všechny potřebné event listenery atd.

Když je někde popsán server-side rendering (SSR) ve SPA aplikacích, je tím většinou myšlen právě tento způsob renderingu s rehydration.

#### Progressive hydration

Problémem rehydratace je velká výpočetní zátěž při načítání stránky. Toto se zejména na výpočetně slabších zařízeních jako jsou chytré telefony projevuje nejvíce. Hlavním problémem je, že stránka na takovémto telefonu vypadá funkčně, ale není interaktivní třeba i po několik sekund.

Způsob, jak tento problém vyřešit je rozdělit tento velký úkol hydratace na mnoho menších. Způsob, jakým to lze udělat, je staticky zanalyzovat celou aplikaci a pro každou komponentu zjistit, jaké má závislosti a jaké události jdoucí z komponenty ovlivní jaké jiné komponenty v aplikaci. Díky tomu jsme aplikaci schopni rozsekat na mnoho malých částí, které jsme schopni dynamicky načítat a spouštět až v případě potřeby. Toto je pro frameworky/knihovny implementačně velice náročný úkol. S progressive hydration se již experimetuje v ReactJS (Markbåge, 2019) a diskutuje v Angularu (Cross, 2016). Velmi pěkně shrnuje problémy a nástrahy implementace progressive hydration v Angularu ve svém komentáři k návrhu implementace Minko Gechev (Gechev, 2020).

#### Partial rehydration

Nehydratuje se celá aplikace najednou, ale jednotlivé komponenty – až v době kdy je např. chceme použít. Experimentální všude, použijeme, pokud zbyde čas

#### Trisomorphic rendering

Server posílá vyrenderované HTML s css, js, javascript zde spouští service worker in advance na pozadí – v browseru již máme nastasrtovanou js aplikaci.

### Static server-side rendering

Během build time vygenerováno statické HTML, JS kompletně odstraněn. Z pohledu kódu ale píšeme normální react aplikaci, používající fetch atd…

### Dynamic server rendering (Turbolinks style)

Klasika HTML Request <-> response, přemažeme komplet <body>, zachováme ale styly + js který je společný pro celou aplikaci.

<https://github.com/turbolinks/turbolinks>

### Dynamic rendering

Jinej content posíláme botům a crawlerům, jiný uživatelům ([www.prerender.io](http://www.prerender.io))

## Metriky měření výkonnosti webu

Změřit výkonnost webové aplikace není zdaleka tak jednoduchý úkol, jak se na první pohled může zdát. Vezměme si například naivní přístup k měření výkonnosti. Měřit budeme čas, za jak dlouho budeme mít kompletní odpověď na náš HTTP požadavek. Taková metrika může být vypovídající pro aplikace renderované na serveru. V momentě, kdy naše aplikace bude napsaná client rendered SPA, dostaneme odpověď vždy téměř okamžitě. V tomto případě se hlavní část práce začne dít až při spuštění javascriptu, který začne přes API stahovat potřebná data, což může trvat dalších několik sekund. Z uživatelského pohledu tedy může serverově renderovaná aplikace působit mnohem výkonněji než SPA aplikace, i přesto že čas odpovědi na HTTP požadavek může být násobně delší.

Jak lze vidět z výše popsaného příkladu, vymyslet metriku, která by věrohodně měřila výkon webové aplikace z uživatelského pohledu, není jednoduchý úkol. Za poslední léta vývoje a měření výkonnosti webových aplikací bylo vymyšleno několik různých metrik, které si v následujících sekcích popíšeme.

Nejdůležitějšími metrikami, podle kterých budeme měřit naše aplikace budou metriky tzv. Core web vitals, kterými jsou LCP, FID, CLS (Google LLC, 2020). Tyto metriky nám přijdou nejvíce reprezentativní pro měření výkonnosti aplikace z uživatelského pohledu. Kromě toho také Google od příštího roku bude ve svém vyhledávacím algoritmu tyto metriky zohledňovat a zvýhodňovat weby, které dosahují v těchto metrikách nižších hodnot (Subramanian‎, 2020).

### Time To First Byte (TTFB)

Time to first byte metrika je dle MDN definován takto:

„Time to First Byte (TTFB) označuje čas mezi začátkem požadavku na stránku a prvním přijatým bytem informace ze serveru. Tento čas zahrnuje DNS lookup, navázaní spojení pomocí TCP handshake a také SSL handshake v případě, že byl požadavek proveden pomocí HTTPS.“ (Mozilla Contributors, 2019)

Metrika TTFB tedy měří hlavně rychlost odpovědi serveru na požadavek klienta. Při čistě serverově renderovaných webových aplikacích můžeme z této metriky vyčíst slabá místa naší aplikace.

„Ukazuje tedy rychlost serveru a backendové části aplikace. A taky samozřejmě rychlost sítě mezi serverem a klientem, tedy měřící aplikací či prohlížečem.“ (Michálek, 2019a)

### First Paint (FP)

„Moment, kdy po kliknutí uživatele prohlížeč vyrenderuje cokoliv, co je vizuálně odlišné od stávající obrazovky.“ (Michálek, 2019d)

### First Contentful Paint (FCP)

### DOM Content Loaded (DCL)

### First Meaningful Paint (FMP)

„Metrika rychlosti webu, která informuje, kdy začne být viditelný primární obsah stránky.“(Michálek, 2019c)

### Speed Index

### Largest Contentful Paint (LCP)

### First Input Delay (FID)

### Cumulative Layout Shift (CLS)

### Time to Interactive (TTI)

„Metrika rychlosti webu, která říká kdy je stránka renderovaná a zároveň schopná spolehlivě reagovat na uživatelský vstup.“(Michálek, 2019b)

### Load

### Lighthouse Performance Score

## Vyhledávače a rendering

### Google

<https://developers.google.com/search/docs/guides/fix-search-javascript>

<https://developers.google.com/search/docs/guides/javascript-seo-basics>

* Lazy loaded images
* Infinite scrolling

### Seznam

### Bing

# Analýza implementační části

## Client rendered

## (Re)hydrated rendering

# Implementace

# Vyhodnocení

# Obrázky, tabulky a programy

Používání tabulek a grafů/obrázků v odborném textu má některá společná pravidla a některá specifická. Tabulky a grafy/obrázky neuvádíme přímo do textu, ale umístíme je buď na samostatné stránky nebo na vyhrazené místo v horní nebo dolní části běžných stránek.

Grafy/obrázky a tabulky se číslují a jsou vybaveny legendou. Legenda má popisovat obsah grafu či tabulky tak podrobně, aby jim čtenář rozuměl bez důkladného studování textu práce.

Na tabulku a graf/obrázek musí být v textu číselný odkaz (lze důrazně doporučit dynamický mechanismus křížových referencí). Na příslušném místě textu pak shrneme ty nejdůležitější závěry, které lze z tabulky či grafu učinit. Text by měl být čitelný a srozumitelný i bez prohlížení tabulek a grafů a tabulky a grafy by měly být srozumitelné i bez podrobné četby textu.

## Obrázky

Několik rad týkajících se obrázků a grafů.

* Graf by měl být vytvořen ve velikosti, v níž bude použit v práci. Zmenšení příliš velkého grafu vede ke špatné čitelnosti popisků. Naopak zvětšení příliš malého grafu vede k nízké grafické kvalitě.
* Osy grafu musí být řádně popsány ve stejném jazyce, v jakém je psána práce (absenci diakritiky lze tolerovat). Kreslíme-li graf hmotnosti proti výšce, nenecháme na nich popisky ht a wt, ale osy popíšeme *Výška [cm]* a *Hmotnost [kg]*. Kreslíme-li graf funkce *h*(*x*), popíšeme osy *x* a *h*(*x*). Každá osa musí mít jasně určenou škálu.
* Chceme-li na dvourozměrném grafu vyznačit velké množství bodů, dáme pozor, aby se neslily do jednolité černé tmy. Je-li bodů mnoho, zmenšíme velikost symbolu, kterým je vykreslujeme, anebo vybereme jen malou část bodů, kterou do grafu zaneseme. Grafy, které obsahují tisíce bodů, dělají problémy hlavně v elektronických dokumentech, protože výrazně zvětšují velikost souborů.
* Budeme-li práci tisknout černobíle, vyhneme se používání barev. Čáry rozlišujeme typem (plná, tečkovaná, čerchovaná, . . .), plochy dostatečně rozdílnými intenzitami šedé nebo šrafováním. Význam jednotlivých typů čar a ploch vysvětlíme buď v textové legendě ke grafu anebo v grafické legendě, která je přímo součástí obrázku.
* Vyhýbejte se bitmapovým obrázkům o nízkém rozlišení a zejména JPEGům (zuby a kompresní artefakty nevypadají na papíře pěkně).

Nadpisy obrázků vkládejte přes *Reference* a *Vložit titulek*. Křížovým odkazem (*Vložení* – *Záložka* a následně *Vložení* – *Křížový odkaz*) se pak na něj odvolejte v textu práce (viz Obr. 1.1). Stejný princip platí pro titulky tabulek a výpisů z programu. U tabulek se popisek vkládá nad ni, popisky obrázků se vkládají pod objekt.

Obr. 1.1 Četnosti různých velikostí bot v populaci žen, mužů a dětí (data ČSÚ, vlastní zpracování)

### Tabulky

U **tabulek** se doporučuje dodržovat následující pravidla:

* Do textu závěrečné práce se nikdy nekopírují tabulky ze statistického softwaru. V typických tabulkách, které tisknou na obrazovku statistické softwary, je rovněž více informací, než je potřeba prezentovat.
* Vyhýbat se svislým linkám. Silnějšími vodorovnými linkami oddělit tabulku od okolního textu včetně legendy, slabšími vodorovnými linkami oddělovat záhlaví sloupců od těla tabulky a jednotlivé části tabulky mezi sebou.
* Neměnit typ, formát a význam obsahu políček v tomtéž sloupci (není dobré do téhož sloupce zapisovat tu průměr, jinde procenta).
* Neopakovat tentýž obsah políček mnohokrát za sebou. Máme-li sloupec *Rozptyl*, který v prvních deseti řádcích obsahuje hodnotu 0*,*5 a v druhých deseti řádcích hodnotu 1*,*5, pak tento sloupec raději zrušíme a vyřešíme to jinak. Například můžeme tabulku rozdělit na dvě nebo do ní vložit popisné řádky, které informují o nějaké proměnné hodnotě opakující se v následujícím oddíle tabulky (např. *„Rozptyl* = 0*,*5*“* a níže *„Rozptyl* = 1*,*5*“*).
* Všechna čísla jsou uvedena s přesností na stejný počet platných číslic. Čísla v tabulce zarovnávat na desetinnou čárku.
* V tabulce je někdy potřebné používat zkratky, které se jinde nevyskytují. Tyto zkratky můžeme vysvětlit v legendě nebo v poznámkách pod tabulkou. Poznámky pod tabulkou můžeme využít i k podrobnějšímu vysvětlení významu některých sloupců nebo hodnot.

Tab. 1.1 Maximální věrohodné odhady v modelu M (data ČSÚ, vlastní zpracování)

|  |  |  |  |
| --- | --- | --- | --- |
| Efekt | Odhad | Směrod. chybaa | P-hodnota |
| Abs. člen | −10,01 | 1,01 | — |
| Pohlaví (muž) | 9,89 | 5,98 | 0,098 |
| Výška (cm) | 0,78 | 0,12 | <0,001 |

*Pozn:*a Směrodatná chyba odhadu metodou Monte Carlo.

V tabulkách je třeba manuálně formátovat text dle následujících pokynů:

* Číselná data jsou v buňkách zarovnána vpravo.
* Pokud hodnota chybí, je uveden znak pomlčka.
* Texty umístěné v buňkách tabulky jsou zarovnány na levý praporek nebo na střed a slova se nedělí.“ (viz Tab. 1.1).

### Programy

Algoritmy, výpisy programů a popis interakce s programy je vhodné odlišit od ostatního

textu (viz Výpis 1.1).

Výpis 5.1 Ukázka formátování krátkého výpisu programového kódu (vlastní zpracování)

1 function prenacteni() {  
2 //fetch polozka budget  
3 var action = "sledovat\_souhrn";  
4 var id\_hry = $('#sledovat\_prubeh\_nazev').attr('rel');  
5 var id\_uzivatele = "<?php echo $\_SESSION['login']['id\_uzivatele']; ?>";  
6   
7 // setInterval(function() {  
8 $.ajax({  
9 url: "ajax/play\_game/kontrola\_ready\_uzivatel.php",  
10 method:"POST",  
11 dataType: "json",  
12 data: {action:action, id\_uzivatele:id\_uzivatele},  
13   
14 success: function (data) {  
15 // $('#rrr').HTML(data[1]);  
16 if(data[0]==true) {  
17 $(".container-log").css({ "display": "none"});  
18 window.clearInterval(prenacteni);  
19 kontrola\_kola();  
20 // $('#rrr').attr('rel', data[0] );  
21 }  
22 },  
23 complete: function (data) {  
24 window.setInterval(function () {  
25 prenacteni();  
26 }, 15000);  
27 }  
28 });  
29 // }, 10000); // In every 30 seconds

### Sazba matematiky

Vzorce vkládejte do textu přes MathType (do Wordu nutné doinstalovat). Vzorce vkládejte a číslujte v záložce *MathType* odkaz *Right-numbered*.



Podrobný popis formátování vzorců a modelů naleznete na intranetu v samostatném dokumentu ***Formátování vzorců a modelů***. 

# Práce s literaturou

K odkazování na literaturu v rámci textu používejte tzv. Harvardský systém (Radváková & al, 2018), nebo číselný systém, který máte podrobně popsán v **dokumentu pro práci s literaturou a dalšími zdroji zpřístupněného na intranetu**. K vytvoření seznamu použité literatury využívejte doporučený nástroj Zotero, jehož instalaci a následné použití naleznete na stránkách [Centra informačních a knihovnických služeb](https://knihovna.vse.cz/wp-content/uploads/page/66/zotero.pdf) VŠE.

Závěr

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean id metus id velit ullamcorper pulvinar. In convallis. Sed convallis magna eu sem. Maecenas lorem. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Ut tempus purus at lorem. Phasellus et lorem id felis nonummy placerat. Mauris elementum mauris vitae tortor. Aliquam id dolor. Vivamus porttitor turpis ac leo. Maecenas fermentum, sem in pharetra pellentesque, velit turpis volutpat ante, in pharetra metus odio a lectus. Phasellus et lorem id felis nonummy placerat.

Použitá literatura

CROSS, Jeff, 2016. feat(DomRenderer): allow partial DOM hydration from pre-rendered content · Issue #13446 · angular/angular. *GitHub* [vid. 20. září 2020]. Získáno z: https://github.com/angular/angular/issues/13446

DHH, 2016. The Majestic Monolith. *Signal v. Noise* [vid. 20. září 2020]. Získáno z: https://m.signalvnoise.com/the-majestic-monolith/

DSHAPS, 2016. Understanding Component-Based Architecture. *Medium* [vid. 20. září 2020]. Získáno z: https://medium.com/@dan.shapiro1210/understanding-component-based-architecture-3ff48ec0c238

GECHEV, Minko, 2020. feat(DomRenderer): allow partial DOM hydration from pre-rendered content · Issue #13446 · angular/angular. *GitHub* [vid. 20. září 2020]. Získáno z: https://github.com/angular/angular/issues/13446

GOOGLE LLC, 2020. *Web Vitals* [vid. 1. srpen 2020]. Získáno z: https://web.dev/vitals/

GRAPHQL FOUNDATION, 2020. *GraphQL: A query language for APIs.* [vid. 20. září 2020]. Získáno z: http://graphql.org/

MALÝ, Martin, 2009. Web Sockets. *Zdroják*. [vid. 20. září 2020]. Získáno z: https://www.zdrojak.cz/clanky/web-sockets/

MARKBÅGE, Sebastian, 2019. Partial Hydration by sebmarkbage · Pull Request #14717 · facebook/react. *GitHub* [vid. 20. září 2020]. Získáno z: https://github.com/facebook/react/pull/14717

MICHÁLEK, Martin, 2019a. Co je „Doba do načtení prvního bajtu“ (aneb „Time To First Byte“ aneb TTFB)? *Vzhůru dolů* [vid. 18. srpen 2020]. Získáno z: https://www.vzhurudolu.cz/prirucka/ttfb

MICHÁLEK, Martin, 2019b. Metrika „Čas do interaktivity“ (Time To Interactive, TTI). *Vzhůru dolů* [vid. 3. srpen 2020]. Získáno z: https://www.vzhurudolu.cz/prirucka/metrika-tti

MICHÁLEK, Martin, 2019c. Metrika „První smysluplné vykreslení“ (First Meaningful Paint, FMP). *Vzhůru dolů* [vid. 3. srpen 2020]. Získáno z: https://www.vzhurudolu.cz/prirucka/metrika-fmp

MICHÁLEK, Martin, 2019d. Událost „První vykreslení“ (First Paint, FP). *Vzhůru dolů* [vid. 3. srpen 2020]. Získáno z: https://www.vzhurudolu.cz/prirucka/metrika-fp

MILLER, Jason a Addy OSMANI, 2019. Rendering on the Web. *Google Developers* [vid. 18. srpen 2020]. Získáno z: https://developers.google.com/web/updates/2019/02/rendering-on-the-web

MOZILLA CONTRIBUTORS, 2019. Time to first byte. *MDN Web Docs* [vid. 1. srpen 2020]. Získáno z: https://developer.mozilla.org/en-US/docs/Glossary/time\_to\_first\_byte

PICHLÍK, Roman, 2017. A REST. *Medium* [vid. 20. září 2020]. Získáno z: https://dagblog.cz/a-rest-c5156313d79e

SUBRAMANIAN‎, Sowmya, 2020. Evaluating page experience for a better web. *Official Google Webmaster Central Blog*. [vid. 18. srpen 2020]. Získáno z: https://webmasters.googleblog.com/2020/05/evaluating-page-experience.html

THE OPEN UNIVERSITY, 2020. An introduction to web applications architecture. *OpenLearn* [vid. 20. září 2020]. Získáno z: https://www.open.edu/openlearn/science-maths-technology/introduction-web-applications-architecture/content-section-0

s

Poznámka: Za koncem Použité literatury nutné nesmazat *konec oddílu*, aby fungovalo různé číslování hlavního textu práce a příloh.

Přílohy

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean id metus id velit ullamcorper pulvinar. In convallis. Sed convallis magna eu sem. Maecenas lorem. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Ut tempus purus at lorem. Phasellus et lorem id felis nonummy placerat. Mauris elementum mauris vitae tortor. Aliquam id dolor. Vivamus porttitor turpis ac leo. Maecenas fermentum, sem in pharetra pellentesque, velit turpis volutpat ante, in pharetra metus odio a lectus. Phasellus et lorem id felis nonummy placerat.

1. Název první přílohy

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean id metus id velit ullamcorper pulvinar. In convallis. Sed convallis magna eu sem. Maecenas lorem. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Ut tempus purus at lorem.

1. Mauris elementum mauris vitae tortor.
2. Aliquam id dolor.
3. Vivamus porttitor turpis ac leo.
4. Phasellus et lorem id felis nonummy placerat.

Maecenas fermentum, sem in pharetra pellentesque, velit turpis volutpat ante, in pharetra metus odio a lectus. Phasellus et lorem id felis nonummy placerat.

1. Název druhé přílohy

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean id metus id velit ullamcorper pulvinar. In convallis. Sed convallis magna eu sem. Maecenas lorem. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Ut tempus purus at lorem. Phasellus et lorem id felis nonummy placerat. Mauris elementum mauris vitae tortor. Aliquam id dolor. Vivamus porttitor turpis ac leo. Maecenas fermentum, sem in pharetra pellentesque, velit turpis volutpat ante, in pharetra metus odio a lectus. Phasellus et lorem id felis nonummy placerat.

1. Pěkný článek popisující specifika monolitických webových aplikací v (DHH, 2016) [↑](#footnote-ref-2)
2. <https://insis.vse.cz/> [↑](#footnote-ref-3)
3. Více do detailu rozebraná specifika client-server architektury lze nalézt např. v (The Open University, 2020) [↑](#footnote-ref-4)
4. Více o REST konceptu v (Pichlík, 2017) [↑](#footnote-ref-5)
5. Více o GraphQL v (GraphQL Foundation, 2020) [↑](#footnote-ref-6)
6. Více o Websocketech např. v (Malý, 2009) [↑](#footnote-ref-7)
7. Pěkné shrnutí specifik component oriented architektury ve webových aplikací v (dshaps, 2016) [↑](#footnote-ref-8)
8. <https://reactjs.org/> [↑](#footnote-ref-9)
9. <https://angular.io/> [↑](#footnote-ref-10)
10. <https://vuejs.org/> [↑](#footnote-ref-11)
11. <https://mail.google.com/> [↑](#footnote-ref-12)
12. <https://nodejs.org/en/> [↑](#footnote-ref-13)